A Temporal Database Mediator for Protocol-Based Decision Support

John H. Nguyen, Yuval Shahar, Samson W. Tu, Amar K. Das, and Mark A. Musen
Section on Medical Informatics, Stanford University School of Medicine,
Stanford, CA 94305-5479

To meet the data-processing requirements for protocol-based decision support, a clinical data-management system must be capable of creating high-level summaries of time-oriented patient data, and of retrieving those summaries in a temporally meaningful fashion. We previously described a temporal-abstraction module (RÉSUMÉ) and a temporal-querying module (Chronus) that can be used together to perform these tasks. These modules had to be coordinated by individual applications, however, to resolve the temporal queries of protocol planners. In this paper, we present a new module that integrates the previous two modules and that provides for their coordination automatically. The new module can be used as a standalone system for retrieving both primitive and abstracted time-oriented data, or can be embedded in a larger computational framework for protocol-based reasoning.

DATA-PROCESSING REQUIREMENTS FOR PROTOCOL-BASED DECISION SUPPORT

The automation of protocol-based decision support involves the generation of recommendations for medical care by a computer program using an electronic patient record and a knowledge base of protocols. Such recommendations are often predicated on the detection of clinically relevant **temporal abstractions** of time-oriented data and **temporal patterns** in the patient record. For example, a health-care provider should modify the standard dose of AZT for a patient if, *during* treatment according to a California Cooperative Treatment Group protocol (CCTG-522), the patient experiences a *second* episode of moderate anemia that has *persisted* for more than 2 weeks. Verification of this condition requires the ability to abstract disjoint episodes of anemia from electronically stored, time-stamped hemoglobin values, and then to select an episode that matches the specified temporal conditions.

The temporal-abstraction and temporal-querying tasks are relevant to any domain in which data are collected and analyzed over time. Protocol-based decision support, in particular, often requires primitive patient data to be summarized into abstract, interval-based concepts that can be retrieved in a consistent and meaningful fashion. Although these tasks are essential and complementary in the automation of protocol-based care, no standard database-query languages currently perform both. To address this limitation, most developers of protocol-based decision support programs have either (1) extended the data-abstraction capabilities of existing data-management systems and stored the resulting abstractions into the database (e.g., Arden Syntax[1]), or (2) provided data-management techniques within the knowledge-based system (e.g., ONCOCIN[2]). We have argued previously that neither approach alone permits the reuse of these capabilities across multiple applications and databases, and that this reuse can be facilitated by two well-encapsulated components.[3]

In this paper, we present a strategy for coupling the temporal-abstraction and temporal-querying tasks in a manner that can address the data-processing requirements of protocol-based decision support. We briefly introduce two modular systems, named **RÉSUMÉ**[4] and **Chronus**,[5] that can support these tasks in a domain-independent manner but that must be coordinated by client applications. To relieve client applications of this requirement, we describe a mediator module, **Tzolkin**,[†] that coordinates automatically the temporal-abstraction and temporal-querying mechanisms of RÉSUMÉ and Chronus.

THE RÉSUMÉ AND CHRONUS MODULES

To avoid the previous incompatibility problems between protocol planners and clinical databases, we have created modular temporal-abstraction and temporal-querying components, the RÉSUMÉ and Chronus systems, respectively. We developed RÉSUMÉ using CLIPS, a C-based production-rule shell. We developed Chronus using a C-based interface to an Open Database Connectivity (ODBC)-compliant relational database. Implementation of these systems as independent components permits the separation of the domain knowledge of a temporal-abstraction module from the data-access methods of underlying relational database systems. This modular

[†] Tzolkin is the Mayan term for the Sun Stone, which served as an accurate representation of calendar time.

1091—8280/97/$5.00

approach maximizes the reusability of our two components with different client applications and databases.

RÉSUMÉ and Chronus provide complementary types of temporal deductions over patient data. RÉSUMÉ, a module based on the **knowledge-based temporal-abstraction (KBTA)** method,[6] uses both general clinical and protocol-specific knowledge to extract from primitive data (in working memory) high-level summaries of a patient's condition over time (such as the abstraction of time-stamped hemoglobin values into intervals of anemia). Chronus provides **TimeLineSQL (TLSQL)**, a general SQL-based data-access language, to make temporal queries on data stored in relational databases. For example, the TLSQL query that retrieves *the first hemoglobin datum acquired before the year 1997 that belongs to patient 123* could be formulated as follows:

```
GRAIN       YEAR
SELECT      FIRST parameter
FROM        lab_results
WHERE       parameter = 'Hemoglobin-value'
            AND patient_id = '123'
WHEN        start_time BEFORE '1997'
```

RÉSUMÉ, unlike Chronus, does not support queries over multiple patients or queries consisting of complex temporal conditions; Chronus, unlike RÉSUMÉ, does not support the identification of intervals that are not stored explicitly in the database. With the complementary actions of these systems, we can provide the temporal data-management services required to automate protocol-based decision support. As we indicated previously, however, the functions of these two modules must be coordinated by client applications.[3]

THE TZOLKIN TEMPORAL DATABASE MEDIATOR

Because the coordination of RÉSUMÉ and Chronus can be difficult for complex knowledge bases, we have developed a single system, called Tzolkin, that combines them. Tzolkin encapsulates the underlying relational patient database, providing access to the data via a query language that can compute abstractions automatically as needed, and can retrieve them in a temporally meaningful fashion. All interactions between the database and knowledge base are completely transparent to the user and the calling process. Such a system is termed a *mediator*,[7] because it serves as a middle layer between the user-oriented processing of applications and the data-manipulation methods of database systems.

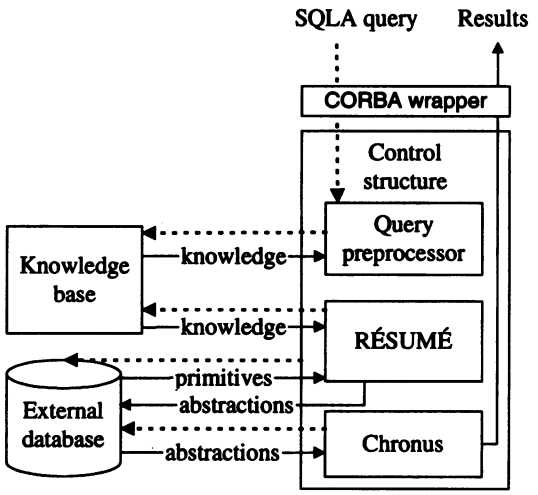

Figure 1: Schematic overview of the Tzolkin architecture, including the interrelationships of the modules. Dashed lines represent queries; solid lines represent the information returned.

To provide this automatic coordination, Tzolkin must (1) identify the requested abstractions in a query statement, (2) determine what domain-specific knowledge and primitive data are required to compute those abstractions, (3) determine where to retrieve these primitives, and (4) invoke the temporal-abstraction and temporal-querying modules with the appropriate information. We have developed a framework based on the Common Object Resource Broker Architecture (CORBA) that satisfies these requirements (Figure 1). Such a framework allows Tzolkin to be used as a standalone system for direct interaction with the care provider or as an embedded component within a larger decision-support framework. The system assumes that an external database exists as the central data repository. Tzolkin comprises the following modules:

1. **RÉSUMÉ** serves as the temporal-abstraction module.
2. **Chronus** serves as the temporal-querying module and as Tzolkin's interface to the underlying relational database. Because Chronus communicates with the database via standard SQL and the ODBC interface, any ODBC-compliant database containing time-stamped intervals of data can be integrated easily into the system.
3. **The query preprocessor** is the module that detects and informs the system of abstractions that need to be computed. The preprocessor also processes requests for Tzolkin's auxiliary services, such as data caching and batch

processing (e.g., computing all possible abstractions given a set of patient data).

4. **The system-control structure** is the top-level module that coordinates the interaction of all other Tzolkin modules. It is responsible for calling each module in the proper order, for ensuring that each module has the necessary information to complete its respective task, and ultimately, for returning the results of a query. By varying the set of modules that is used and the order in which the modules are invoked, the system-control structure can evaluate three different query classes. These classes will be discussed in the following section.

In addition to these four modules, Tzolkin requires an external knowledge base to provide domain- and application-specific knowledge to the system (e.g., knowledge regarding the primitives that are needed to compute an abstraction.) This knowledge defines the data requirements of each module and ensures that all abstractions from RÉSUMÉ are computed in a manner that is semantically consistent with the clinical protocol.

The Query-Evaluation Strategy

The interface to Tzolkin is a query language named **SQL for Abstractions (SQLA)**. SQLA is similar to the Chronus TLSQL language, but provides an important semantic extension. Whereas TLSQL cannot retrieve data unless they are stored explicitly in the database, SQLA can retrieve data that are either stored in the database as primitives or defined in the knowledge base as abstractions of these primitives. The semantics of the SQLA language are, therefore, defined by both the KBTA method and the relational model. Thus, SQLA allows the user to query the database for high-level summaries of the archived data.

Only one syntactic addition to the TLSQL language is necessary in order to provide the semantic extension described. Because primitive clinical data can have multiple context-dependent interpretations (e.g., the classification of a hemoglobin value as LOW in one context and NORMAL in another), the SQLA query must specify the desired interpretation context when requesting abstractions. Thus, we have defined a new CONTEXT clause.

Using SQLA, Tzolkin can process three general classes of queries: those that request (1) abstractions for a single patient, (2) abstractions for multiple patients, and (3) only primitive data (Figure 2). The strategies required to process each of these classes differ only in the set of Tzolkin modules that is used and the order in which the modules are invoked. We will therefore present the detailed evaluation strategy

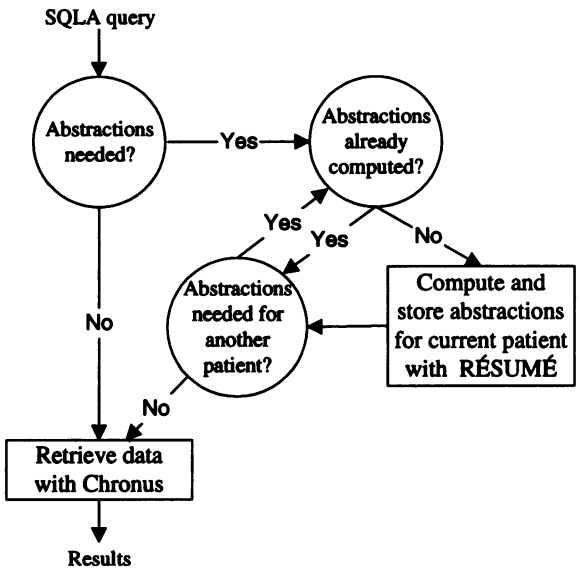

Figure 2: The general query-evaluation strategy. Using this algorithm, Tzolkin can resolve queries that request either primitive or abstracted data for one or more patients.

for the first query class, and then discuss briefly the differences between the other two and the first.

Returning to our introductory example, a protocol planner might ask *whether patient 123 has experienced a second episode of moderate anemia that has persisted for more than 2 weeks during therapy with the CCTG-522 protocol.* The SQLA query requests abstractions for a single patient and could be formulated as follows:

GRAIN	WEEK
CONTEXT	'CCTG-522-Therapy'
SELECT	SECOND problem-name
FROM	patient-problems-view
WHERE	problem-name = 'moderate-anemia'
	AND patient-id = '123'
WHEN	DURATION > '2'

Tzolkin processes this query as follows:

1. On receiving this query, the system-control structure calls the query preprocessor, whose task it is to evaluate the need for performing temporal abstraction.
2. After generating a list of the nonkeyword tokens in the query (e.g., 'moderate-anemia'), the preprocessor must determine whether any of these tokens is defined in the knowledge base as an abstraction of primitive data. The preprocessor finds that 'moderate-anemia' refers to information that can be abstracted from primitives in database, and the CONTEXT

clause indicates that these abstractions should be computed within the context of 'CCTG-522-Therapy'. This information is returned to the system-control structure.

3. The system-control structure now must determine what knowledge and data RÉSUMÉ needs to compute the abstractions identified in step 2. It searches the knowledge base and finds that 'moderate-anemia' is abstracted from 'hemoglobin-value', which can be found in the database 'lab-results' table. It also finds that the event 'CCTG522-START' must be loaded to frame the generated abstractions within the context of therapy with CCTG-522.
4. Using the information from step 3, the system-control structure proceeds to retrieve the 'hemoglobin-value' data directly from the database (using SQL) for patient '123'. It also retrieves the 'CCTG522-START' event from the database and the mapping required for these data to be loaded properly into RÉSUMÉ's working memory.
5. The system-control structure loads these information into RÉSUMÉ using the database-to-RÉSUMÉ mapping rules stored in the knowledge base, thereby activating the RÉSUMÉ abstraction mechanisms.
6. RÉSUMÉ then computes the abstractions. The generated abstractions are stored, with their interval-based time stamps, into the 'patient-problems-view' table of the clinical database. These abstractions are not stored beyond the current session unless so requested by the user.
7. The original query is passed to Chronus, which processes the temporal conditions imposed by the query and returns the appropriate tuples.

Three Tzolkin modules eliminate the need for coordination of Chronus and RÉSUMÉ by individual applications. The query preprocessor detects abstractions that need to be computed. The knowledge base informs the system of which data are required and where they are stored. The system-control structure directs the query-evaluation process, loading the appropriate data and invoking the appropriate modules.

With this strategy, Tzolkin can answer queries requiring abstractions for a single patient, performing the necessary computations transparently. It can use simple variations on this approach to evaluate the two other classes of queries that we described. The second class that we described requests abstracted data for multiple patients. (Such a query might be used, e.g., to identify all patients in the database who meet the eligibility criteria for a particular protocol.) If this type of query is issued to the system, Tzolkin executes the process sequentially for each individual set of patient data. This strategy ensures the proper partitioning of patient-specific data while the abstractions are computed. Finally, the third class of queries requests only primitive patient data from the database. If the query preprocessor does not detect a need for abstractions, the query is simply passed to the Chronus module for TLSQL-style retrieval (i.e., only step 7 of the outlined evaluation strategy is executed).

Tzolkin Batch Computation

In addition to coordinating RÉSUMÉ and Chronus, the integrated Tzolkin architecture provides a batch-computation feature that can optimize system use in certain situations. **Batch computation** involves the generation and storage of all possible abstractions for a given set of data in advance of the their use. Thus, at run time, only the Chronus component is invoked to process the temporal conditions of the query. This feature may be useful in applications where the data set is reasonably static (e.g., during a therapy session where the identity of the patient and all relevant data are known in advance.) When a batch command has been issued and completed for a particular data set, the results will be cached in the database until they are explicitly deleted by the user. Until this deletion, all requests for these abstractions will be referred directly to the database, rather than evaluated in the regular manner. This feature is useful from the perspective of efficiency.

DISCUSSION

We have designed and fully implemented a temporal-database mediator that is capable of answering the abstract, time-oriented queries typically encountered in protocol-based decision support. Tzolkin integrates the RÉSUMÉ and Chronus systems into a single module that can mediate queries to an external relational database. By separating clinical data from clinical domain knowledge, and by making few assumptions about the nature of the external clinical database, we have avoided the incompatibility problems of earlier approaches. Tzolkin can function in two capacities: as a standalone system for use by the care provider who wishes to query and visualize the clinical data directly (in primitive or abstracted form) and as a mediator module embedded within a larger decision-support architecture. We have successfully embedded Tzolkin within such an architecture, named **EON**[8] (Figure 3). We have evaluated Tzolkin within this

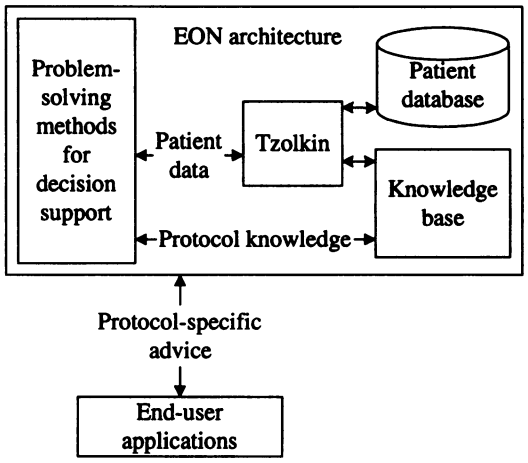

Figure 3: Schematic overview of the EON decision support architecture. Within this framework, Tzolkin mediates all access to the patient database, providing temporal-abstraction and temporal-querying services to the problem-solving modules.

framework and as a standalone system to resolve the three types of temporal queries described. Within our domain of evaluation — AIDS therapy planning — Tzolkin has, informally, been able to meet the data-processing requirements of protocol planners. Future work will include a formal evaluation of Tzolkin's capabilities.

The mediator approach to temporal-data management is novel, and thus raises new problems. One such problem is the difficulty in evaluating the general time complexity of a system that uses both rule-based and database methods; the basic computational operation is difficult to define. We recognize, however, that certain complex queries can be optimized, and we have implemented one feature to do this (batch computation).

Our optimization strategy, itself, raises an important issue: the defeasibility of the computed temporal-abstractions. The RÉSUMÉ system uses a truth-maintenance system that allows specific temporal abstractions to be withdrawn from working memory when new, contradictory data arrive. This truth-maintenance system currently does not extend to the external database. To avoid this problem of nonmonotonicity, Tzolkin computes all abstractions based on the content of the database at the moment that the query is evaluated. Furthermore, it does not save the computed abstractions between queries unless so indicated by the user. Consequently, without a truth-maintenance system at the database level, caching features such as batch computation are useful only when used with a data set that is guaranteed to be consistent over time (an unlikely situation in clinical databases). Several possible solutions have been proposed, such as the use of **temporal checkpoints** beyond which previously computed abstractions are available for retrieval but not for modification.[9] We are currently investigating the most appropriate solutions to nonmonotonicity and optimization.

Acknowledgments

This work has been supported by grants LM05708 and LM06245 from the National Library of Medicine and grants IRI-9528444 and IRI-9257578 from the National Science Foundation. We are grateful to Lyn Dupré for editing this manuscript.

References

1. Hripcsak G, Ludemann P, Pryor TA, Weigertz OB, Clayton PD. Rationale for the Arden syntax. *Computers and Biomedical Research*, 1994; 7(4):291-324.
2. Kahn MG, Fagan, LM, Tu S. Extensions to the time-oriented database model to support temporal reasoning in medical expert systems. *Methods of Information in Medicine*, 1991; 30:4-14
3. Das AK, Shahar Y, Tu SW, Musen MA. A temporal-abstraction mediator for protocol-based decision-support systems. In Ozbolt JG (ed.), *Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care*, Washington, D.C., Hanley & Belfus, 1994; 320-324.
4. Shahar Y, Musen MA. Knowledge-based temporal abstraction in clinical domains. *Artificial Intelligence in Medicine*, 1996; 8:267-298.
5. Das AK, Musen MA. A temporal query system for protocol-directed decision support. *Methods of Information in Medicine*, 1994; 33(4):358-370.
6. Shahar Y. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 1997; 90:79-133.
7. Wiederhold G. Mediators in the architecture of future information systems. *IEEE Computer*, 1992; 25:38-50.
8. Musen MA, Tu SW, Das AK, Shahar Y. EON: A component-based approach to automation of protocol-directed therapy. *Journal of the American Medical Informatics Association*, 1996; 3(6): 367-388.
9. Combi C, Shahar Y. Temporal reasoning and temporal data maintenance in medicine: issues and challenges. *Computers in Biology and Medicine* (in press).